

impara elettronica digitale

...e costruisci il tuo **LABORATORIO DIGITALE**

6,90 €



66



Peruzzo & C.

**TOTALMENTE
PROGRAMMABILE!!!**



Direttore responsabile:
ALBERTO PERUZZO
Direttore Grandi Opere:
GIORGIO VERCELLINI
Consulenza tecnica
e traduzioni:
CONSULCOMP S.n.c.
Pianificazione tecnica
LEONARDO PITTON

Direzione, Redazione, Amministrazione: viale Ercole Marelli 165, Tel. 02/242021, 20099 Sesto San Giovanni (Mi). Pubblicazione settimanale. Registrazione del Tribunale di Monza n. 1738 del 26/05/2004. Spedizione in abbonamento postale gr. II/70; autorizzazione delle Poste di Milano n. 163464 del 13/2/1963. Stampa: Grafiche Porpora s.r.l., Cernusco S/N (MI). Distribuzione SO.DI.P. S.p.A., Cinisello Balsamo (MI).

© 2004 F&G EDITORES, S.A.
© 2005 PERUZZO & C. s.r.l. Tutti i diritti sono riservati. Nessuna parte di questa pubblicazione può essere riprodotta, archiviata su sistema recuperabile o trasmessa, in ogni forma e con ogni mezzo, in mancanza di autorizzazione scritta della casa editrice. La casa editrice si riserva la facoltà di modificare il prezzo di copertina nel corso della pubblicazione, se costretta da mutate condizioni di mercato.

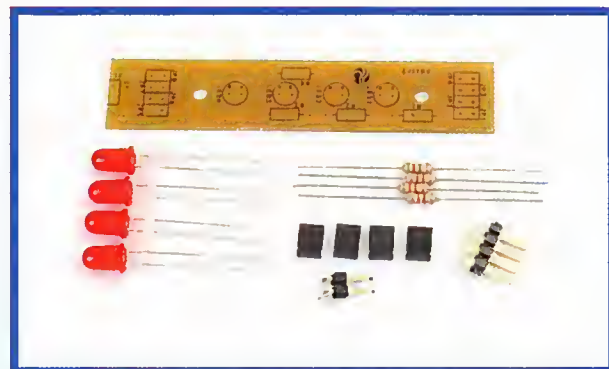
"ELETTRONICA DIGITALE"
si compone di
70 fascicoli settimanali
da suddividere
in 2 raccoglitori.

RICHIESTA DI NUMERI ARRETRATI.
Per ulteriori informazioni, telefonare dal lunedì al venerdì ore 9.30-12.30 all'ufficio arretrati tel. 02/242021. Se vi mancano dei fascicoli o dei raccoglitori per completare l'opera, e non li trovate presso il vostro edicolante, potrete riceverli a domicilio rivolgendovi direttamente alla casa editrice. Basterà compilare e spedire un bollettino di conto corrente postale a PERUZZO & C. s.r.l., Ufficio Arretrati, viale Marelli 165, 20099 Sesto San Giovanni (MI). Il nostro numero di c/c postale è 42980201. L'importo da versare sarà pari al prezzo dei fascicoli o dei raccoglitori richiesti, più le spese di spedizione € 3,10 per pacco. Qualora il numero dei fascicoli o dei raccoglitori sia tale da superare il prezzo globale di € 25,82 e non superiore a € 51,65, l'invio avverrà per pacco assicurato e le spese di spedizione ammontaranno a € 6,20. La spesa sarà di € 9,81 da € 51,65 a € 103,29; di € 12,39 da € 103,29 a € 154,94; di € 14,98 da € 154,94 a € 206,58; di € 16,53 da € 206,58 in su. Attenzione: ai fascicoli arretrati, trascorse dodici settimane dalla loro distribuzione in edicola, viene applicato un sovrapprezzo di € 0,52, che andrà pertanto aggiunto all'importo da pagare. Non vengono effettuate spedizioni contrassegno. Gli arretrati di fascicoli e raccoglitori saranno disponibili per un anno dal completamento dell'opera. **IMPORTANTE:** è assolutamente necessario specificare sul bollettino di c/c postale, nello spazio riservato alla causale del versamento, il titolo dell'opera nonché il numero dei fascicoli e dei raccoglitori che volete ricevere.

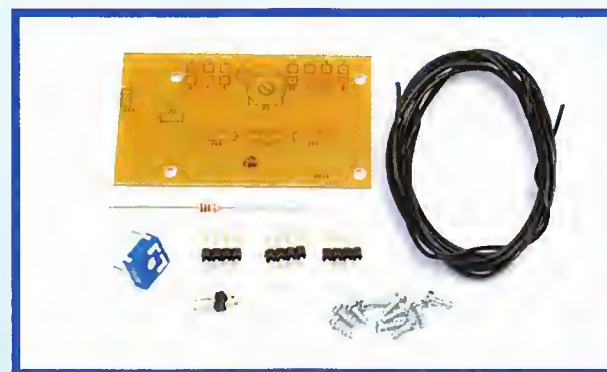
impara elettronica digitale

IN REGALO in questo fascicolo

- 1 Scheda DG11r1
- 4 LED rossi
- 1 Connettore da c.s. maschio, diritto, da 2x4 vie
- 1 Connettore da c.s. maschio, a 90°, a 2 vie
- 4 Resistenze da 820 Ohm 5% 1/4 W
- 4 Ponticelli isolati neri



IN REGALO nel prossimo fascicolo



- 1 Scheda DG14
- 3 Connettori da c.s. diritti maschi a 4 vie
- 1 Potenziometro di regolazione, verticale, da 10 K
- 1 Resistenza da 10 K 5% 1/4 W
- 1 Connettore da c.s. a 90°, maschio a 2 vie
- 10 Viti
- 1 Pezzo di filo nero flessibile

COME RACCOGLIERE E SUDDIVIDERE L'OPERA NELLE 4 SEZIONI

L'Opera è composta da 4 sezioni identificabili dalle fasce colorate, come indicato sotto. Le schede di ciascun fascicolo andranno suddivise nelle sezioni indicate e raccolte nell'apposito raccoglitore, che troverai presto in edicola. Per il momento, ti consigliamo di suddividere le sezioni in altrettante cartellette, in attesa di poterle collocare nel raccoglitore. A prima vista, alcuni numeri di pagina ti potranno sembrare ripetuti o sbagliati. Non è così: ciascuno fa parte di sezioni differenti e rispecchia l'ordine secondo cui raccogliere le schede. Per eventuali domande di tipo tecnico scrivere al seguente indirizzo e-mail: elettronicadigitale@microrobots.it

Hardware Montaggio e prove del laboratorio

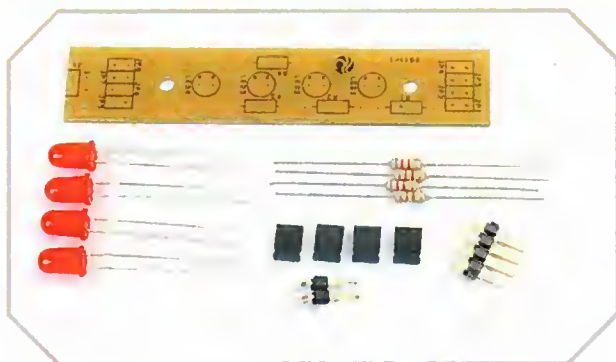
Digitale di base Esercizi con i circuiti digitali

Digitale avanzato Esercizi con i circuiti sequenziali

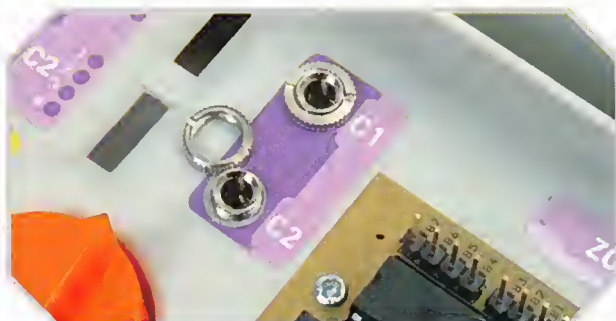
Microcontroller Esercizi con i microcontroller



Connettori ausiliari e matrice dei LED



Componenti allegati a questo fascicolo.



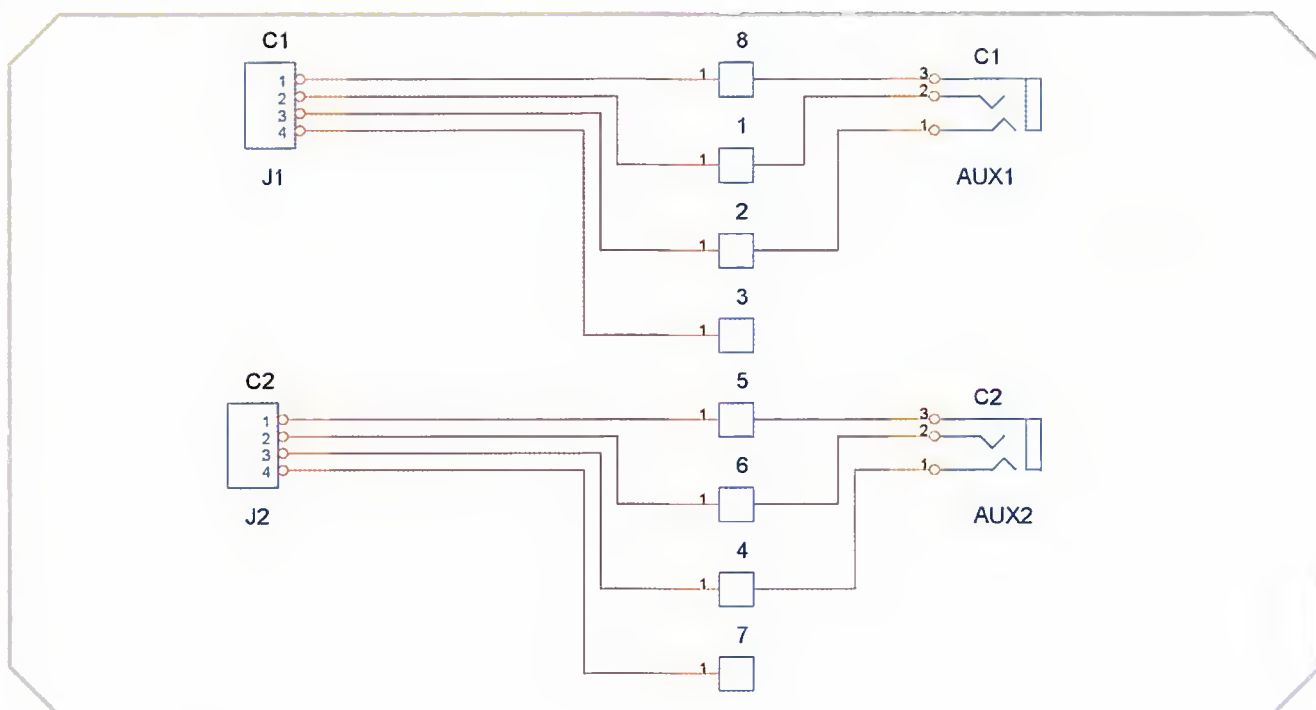
Connettori ausiliari C1 e C2.

Con questo fascicolo viene fornito il materiale necessario per completare il quarto modulo della matrice dei LED.

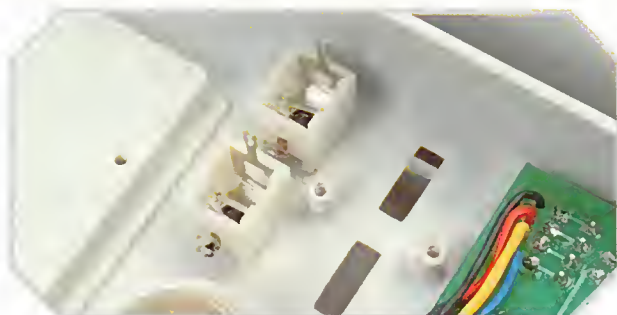
Ricordate che uno dei connettori è già stato fornito con il fascicolo 57. Si monterà e si installerà la scheda DG10, corrispondente ai connettori ausiliari C1 e C2, si realizzerà anche la stessa operazione con la scheda DG11 ora fornita.

Connettori ausiliari

I connettori ausiliari C1 e C2 si utilizzano per facilitare i collegamenti di strumenti e accessori audio che utilizzano connettori concentrici per femmine tipo jack, questo tipo di connettore è molto utilizzato nelle schede audio dei computer e negli altoparlanti e microfoni che si collegano a esse. Il collegamento tra i connettori circolari e il resto del laboratorio si



Schema elettrico.



Vista interna dei connettori C1 e C2.



Scheda DG10 completata.

realizza tramite dei connettori maschi diritti a quattro vie siglati anche loro come C1 e C2, utilizzando dei cavetti terminati su connettori a quattro vie e il sistema di collegamento del laboratorio. Questi connettori sono del tipo jack stereo da 3,5 millimetri e si inseriscono nei due fori circolari siglati come C1 e C2. La loro sede è piuttosto precisa e questo facilita il montaggio impedendo la rotazione e mantenendoli ben fissati; dopo averli inseriti avvieremo i due controdadi di fissaggio.

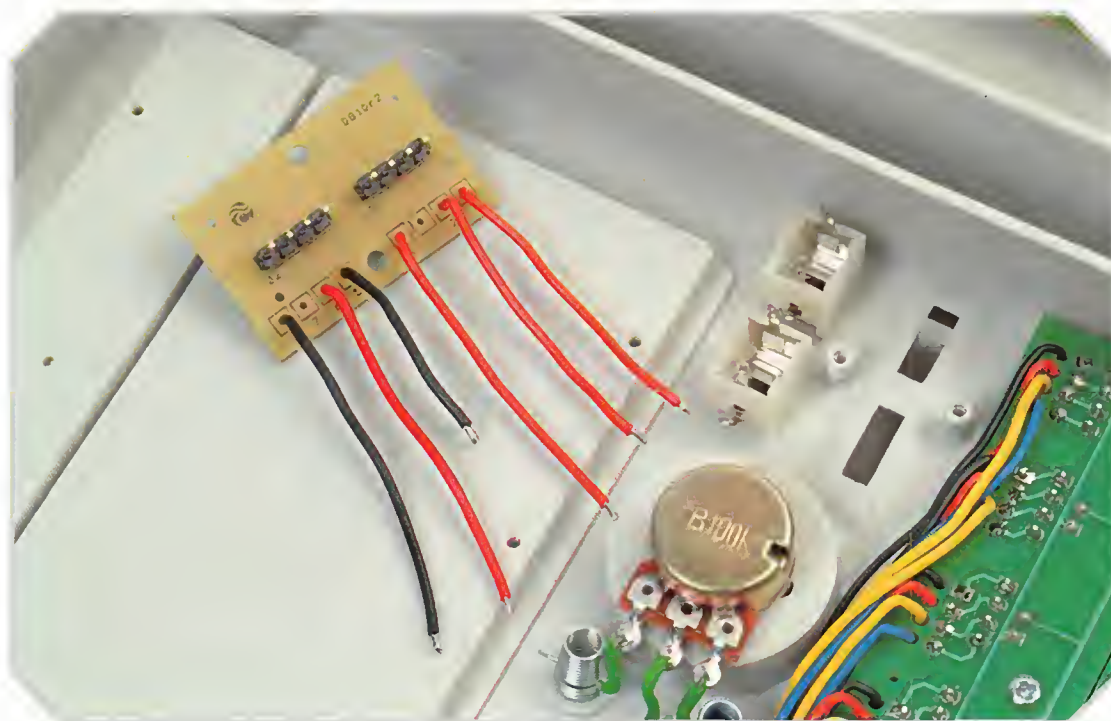
Montaggio della scheda DG10

Il montaggio di questa scheda è semplice, dato che è sufficiente saldare i due connettori con i terminali diritti maschi a quattro vie, con l'unica precauzione di farli rimanere perpendicolari al circuito stampato.

Collegamenti di DG10

Il collegamento di questa scheda si esegue seguendo lo schema elettrico. Osservate i terminali 3 e 7 non sono utilizzati e quelli siglati con 8 e 5 corrispondono alla massa.

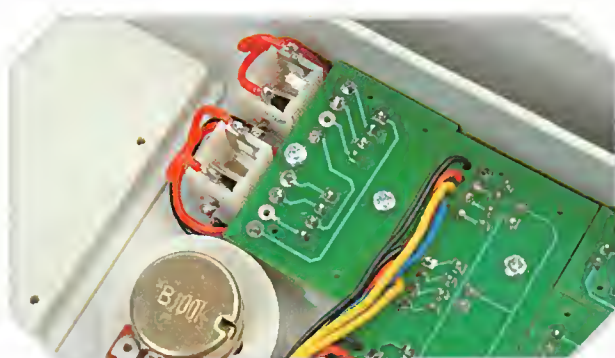
I connettori 8 e 5 si collegano con un filo nero da 5 e da 3 cm di lunghezza rispettivamente, mentre i connettori 1, 2, 6 e 4 si collegano con un filo di colore rosso da 4,5 cm di lun-



Scheda DG10 con i suoi fili.



Dettagli del collegamento.



Scheda DG10 installata.

ghezza per i primi due, e da 5 cm per gli ultimi. Dopo aver saldato questi fili, avvicineremo la scheda alla sua posizione definitiva e li salderemo ai connettori nel modo che possiamo vedere nelle fotografie.

Installazione della scheda

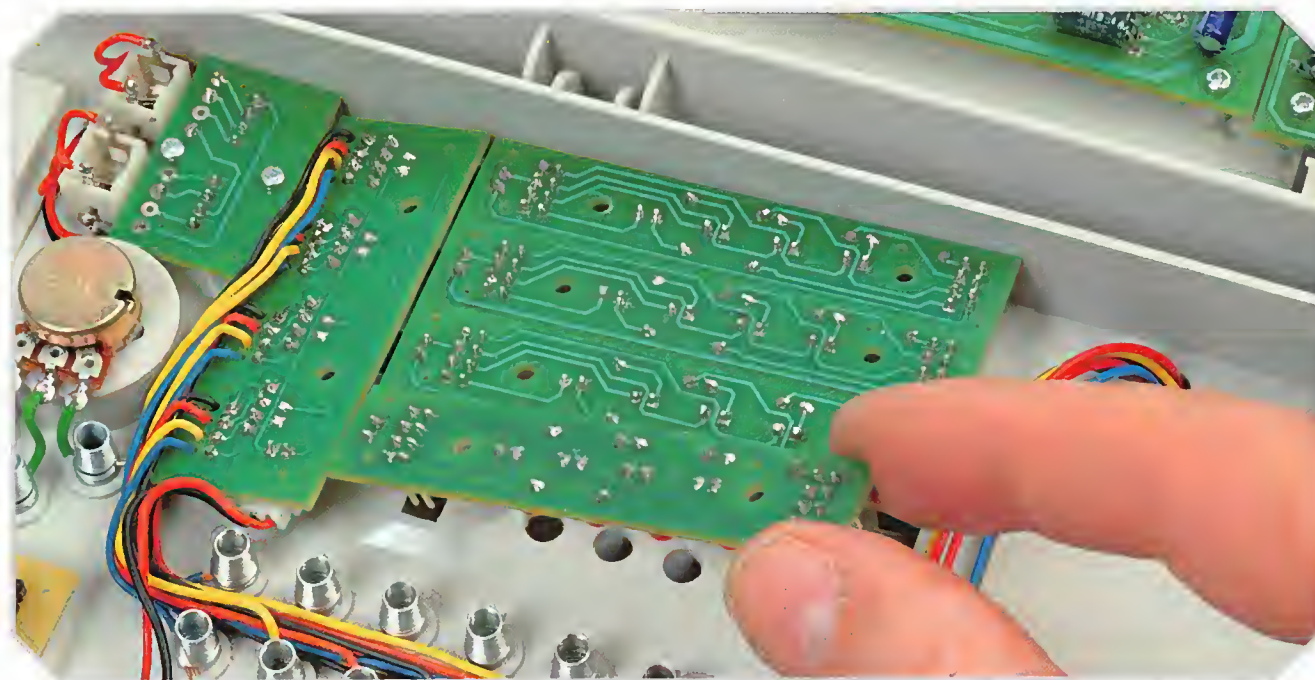
I due connettori maschi a quattro vie della scheda devono rimanere centrati nelle due sedi rettangolari predisposte nel pannello frontale, siglate come C1 e C2. In questo modo, guardando dalla parte interna, rimarranno allineati i fori della scheda e le due colonne dove si fisseranno le due viti che sono già state fornite.

Quarta scheda dei LED

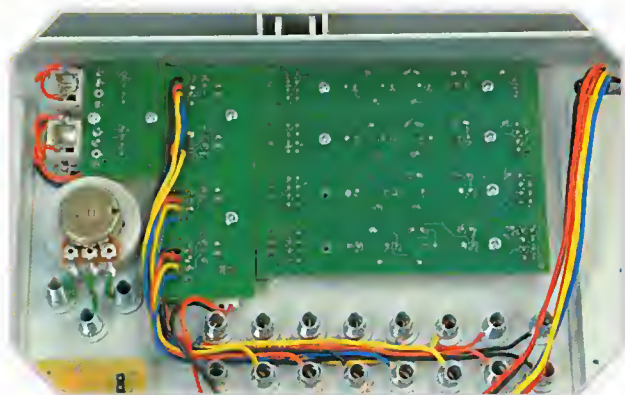
Il montaggio dei componenti sulla scheda DG11, che è identica alle precedenti, è già stato spiegato in precedenza, quindi non lo ripeteremo.

Matrice dei LED

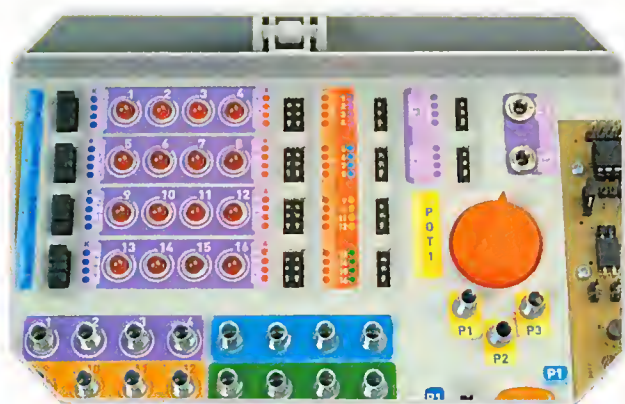
Con questa quarta scheda DG11 possiamo terminare il montaggio della matrice dei LED. Il montaggio si eseguirà nel modo seguente: colleghiamo tutti i tipi di alimentazione, to-



Inserimento della quarta scheda DG11.



Scheda della matrice dei LED.



Vista della matrice dei LED dall'esterno.

gliando anche le pile, rimuoviamo tutte le viti delle schede DG11 installate e quelle della DG12, solleviamo l'insieme inclinandolo a partire dalla scheda DG12 e alzando la parte più lontana dalla scheda, in modo che questa si sollevi meno e permetta che i LED fuoriescano dalle loro sedi.

In questo modo faciliteremo il collegamento della scheda DG11 sul connettore rimasto libero della DG12. Infine abbasseremo tutto l'insieme verificando che i 16 LED fuoriescano così come i loro connettori.

È conveniente inserire qualche ponticello per facilitare la centratura dei connettori nei fori del pannello.

Inseriremo le due viti della scheda DG12, le quattro delle schede DG11 dal lato più lontano dalla DG12, e le due dal lato più vicino, in quanto devono essere ancora fornite due viti.

Prova

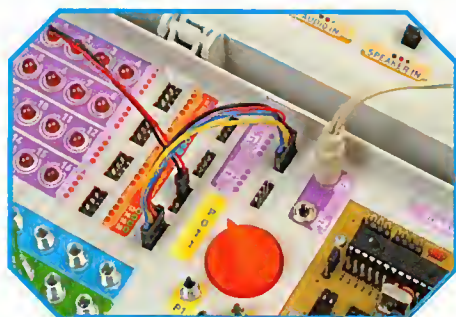
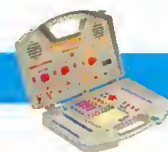
La prova si esegue nel modo spiegato a suo tempo. Per iniziare si tolgono tutti i ponticelli degli anodi e dei catodi e si collega l'alimentazione, solo i 5 volt.

Si inseriscono i ponticelli corrispondenti all'anodo e al catodo del LED 1, il quale si deve illuminare. Sposteremo successivamente i ponticelli da LED in LED fino ad arrivare al 16, verificando così il funzionamento di tutti.



Vista generale del laboratorio.

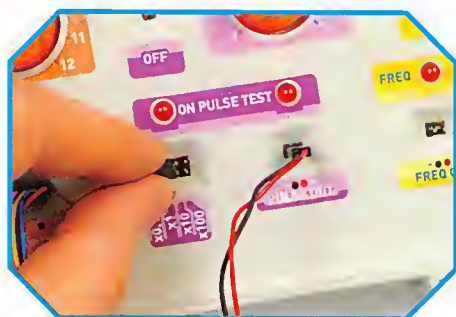
Schema elettrico.



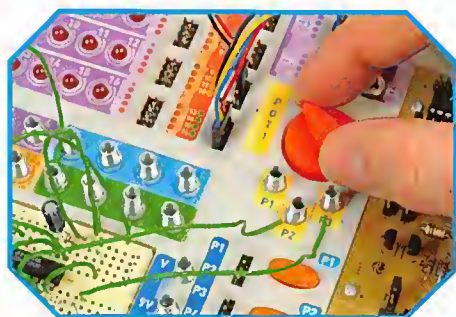
Collegamento degli altoparlanti multimediali.



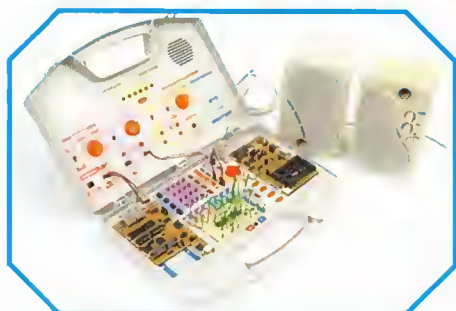
Collegamento del generatore di impulsi.



Dettaglio del montaggio.



Con POT 1 si cambia la frequenza del suono.



Vista completa dell'esperimento.

Montaggio

Il montaggio da eseguire sulla scheda Bread Board è semplice in quanto utilizza pochi componenti. Il generatore di impulsi si collega tramite un cavetto a due conduttori tra l'uscita PULSE OUT e i connettori corrispondenti alle molle 9 e 10. Il connettore maschio C1 a quattro vie si collega con un cavetto terminato su due connettori a quattro contatti inserito su questo connettore e quello corrispondente alle molle dalla 13 alla 16, anche se si utilizzano soltanto le prime tre, e infine il sistema di altoparlanti amplificati si collega con il suo connettore al jack circolare siglato con C1 sul pannello principale del laboratorio. È necessario verificare che si tratti di un connettore stereo. Questo circuito si alimenta a 5 volt e l'uscita manca di componente continua, dato che viene eliminata inserendo i condensatori C4 e C5.

Funzionamento

Dopo aver verificato tutti i collegamenti si posiziona il comando del potenziometro POT 1 e quello del generatore di impulsi all'incirca a metà della loro corsa e si inserisce un ponticello sul generatore di impulsi in una qualsiasi delle sue quattro posizioni. Il volume dello strumento da provare deve essere molto basso all'inizio della prova, e il commutatore PULSE deve essere su ON in modo che riceva alimentazione.

Dobbiamo udire un suono alternato su ogni altoparlante la cui frequenza si regola con POT 1. La cadenza con cui cambia il suono da un altoparlante all'altro si apprezzerà tanto meglio quanto più bassa sarà la frequenza del generatore di impulsi, provate a ruotare il comando di questo generatore e a cambiare la posizione del ponticello.

LISTA DEI COMPONENTI

U1	Circuito integrato 4027
U2	Circuito integrato 4093
R2	Resistenza 1K8 (marrone, grigio, rosso)
R3	Resistenza 100 K (marrone, nero, giallo)
C1	Condensatore 100 nF
C3	Condensatore 22 nF
C4, C5	Condensatore 10 μ F elettrolitico



Primo programma con il PICBASIC PLUS LITE

Dopo aver analizzato separatamente il software PicBasic Plus Lite e la struttura di programmazione di questo ambiente, li uniremo studiandoli attraverso i programmi di esempio che sono allegati al software stesso.

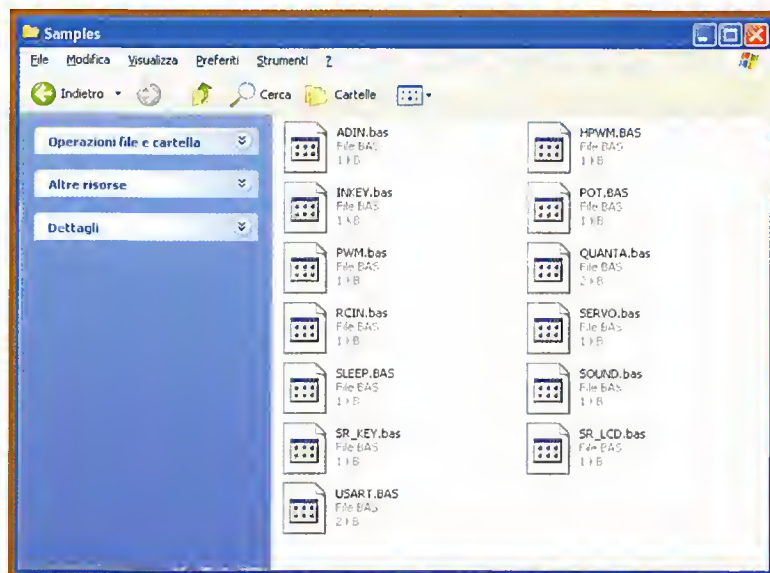
Esercizi di esempio

All'interno della cartella che crea il programma PicBasic Plus Lite durante l'installazione, esiste una directory chiamata "Samples" dove potremo trovare alcuni programmi di esempio. I file di codice o programmi realizzati con questo software hanno come estensione ".bas". Quando salviamo un codice realizzato in questo ambiente di programmazione il file avrà l'estensione ".bas", ma se lo compiliamo viene creata una serie di file che abbiamo già visto altre volte: in assembler, in codice macchina, ecc.

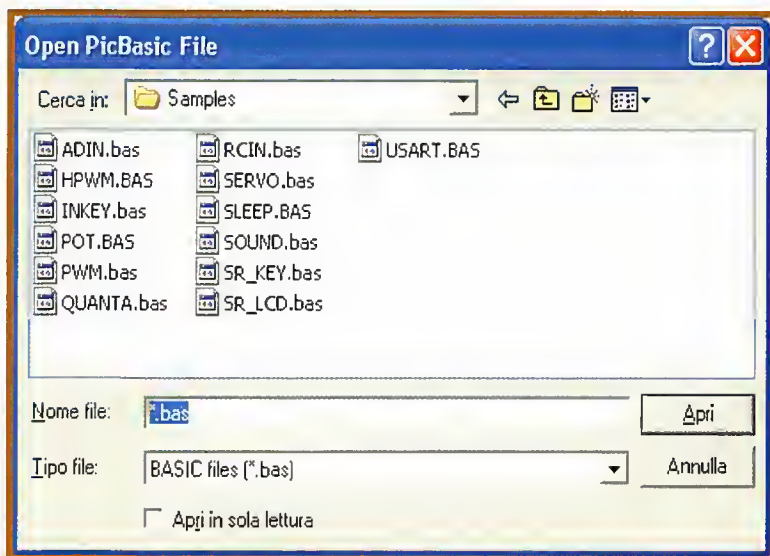
Nella figura possiamo verificare che i file si trovano nella cartella "Samples" e hanno nomi che ci permettono di capire a cosa sono riferiti senza doverli aprire e vederne il contenuto.

Carichiamo un esempio sul PicBasic Plus Lite

Con il programma PicBasic Plus Lite aperto selezioniamo l'opzione Open nella barra degli strumenti o dal menù File. Facendo questo appare una finestra in cui è riportato il contenuto della cartella "Samples". Scegliete il primo dei file "ADIN.bas" e caricatelo. Adesso nell'editor è stato caricato il programma selezionato e possiamo vedere che nella prima linea viene spiegato, mediante un commento, ciò che esegue il programma. Questo programma legge il valore analogico che ha il pin 0 della porta A e lo visualizza.



Cartella "Samples", dove troveremo gli esercizi di esempio.



Selezionando l'opzione Open appare questa finestra.



Carichiamo il programma "ADIN.bas".

verrà registrato il valore di ingresso, in Value convertiranno questo valore, in Volts scriveremo il valore in Volt e in Millivolts il valore in millivolt. Come vedremo, dichiarando le variabili si definisce anche il tipo o la dimensione delle stesse (bit, byte o word).

Create le variabili dovremo definire come ingressi o uscite i pin delle porte del PIC che utilizzeremo. Come abbiamo già visto, possiamo fare

```

- PICBASIC PLUS LITE V1.10 - C:\Programmi\VBPLUS_LITE\Samples\ADIN.bas
File Edit Compile Options Help

' Read channel 0 of on-board ADC and display the results
Device 16F877              ' Choose the 16F877
Declare XTAL 20            ' We'll use a 20MHz Xtal
Declare Rscout_Pin PortB.0 ' Serial out pin is PortB bit-0
Declare ADIN_TAD FPC       ' Choose the RC osc for ADC samples
Declare ADIN_STIME 100     ' Allow 100us for charge time

Dim Raw as Word
Dim Value as Word
Dim Volts as Byte
Dim Millivolts as Word

Rscout Cls                ' Clear the serial LCD
TrisA.0 = 1               ' Setup bit-0 of PortA as an input
ADCON1 = %10000000       ' Right justified result in ADRESL and ADRESH

Again: Raw = ADIN 0       ' Read the ADC
Rscout at 2,1,"RAW= ",@Raw," " ' Display the RAW data
Value = 489 * (Raw / 10)  ' Quantize the result
Volts = Value / 10000
Millivolts = (Value // 10000) / 100
Rscout at 1,1,dec1 Volts,".",dec2 Millivolts,"V " ' Display the result
Goto Again                ' Do it forever
  
```

Il programma ha una struttura familiare. Dopo il commento di inizio selezioniamo il PIC con cui vogliamo lavorare (16F877) e definiamo la frequenza con la quale funzionerà il microcontroller (20 MHz). Proseguiamo definendo i dispositivi esterni configurando il pin 0 della porta B come pin di uscita seriale verso l'LCD, l'oscillatore con cui lavorerà il convertitore analogico-digitale (RC) e impostando un tempo di carica al convertitore di 100 µs.

Continuiamo dichiarando le variabili che utilizzeremo all'interno del programma. In Raw

questo utilizzando l'istruzione DEFINE o direttamente assegnando un valore logico (0 o 1) al bit equivalente del registro associato TRISx. All'interno del programma che stiamo considerando viene fatto direttamente mediante l'istruzione "TrisA.0=1". È necessario configurare anche il registro ADCON1 per fare in modo che questo pin, definito come ingresso, sia di tipo analogico. Il ciclo che deve acquisire il valore dell'ingresso e mandarlo all'LCD in modo che lo visualizzi, inizia a partire dall'etichetta "Again". Si assegna alla prima variabile la lettura del pin di ingresso, lo visualizziamo e vediamo le operazioni sul valore per convertirlo in volt e millivolt e visualizziamo i risultati. Infine, saltiamo all'etichetta e ripetiamo il ciclo.

Istruzioni

I programmi in Basic fanno in modo molto semplice ciò che

	Bit	Byte	Word
	1 bit	8 bit	16 bit
Raw		X	X
Value			X
Volts		X	
Millivolts			X

Tipo di variabili.

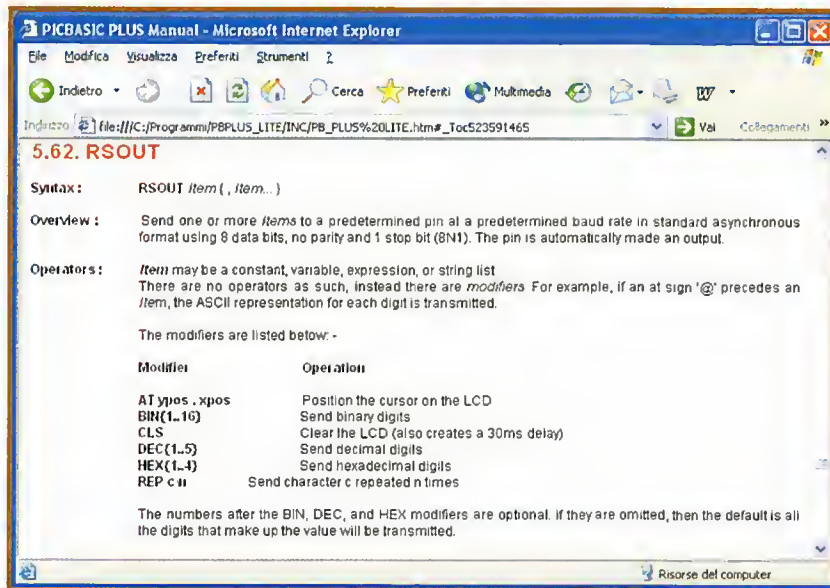
```

- PICBASIC PLUS LITE V1.10 - C:\Programmi\VBPLUS_LITE
Assembler Listing

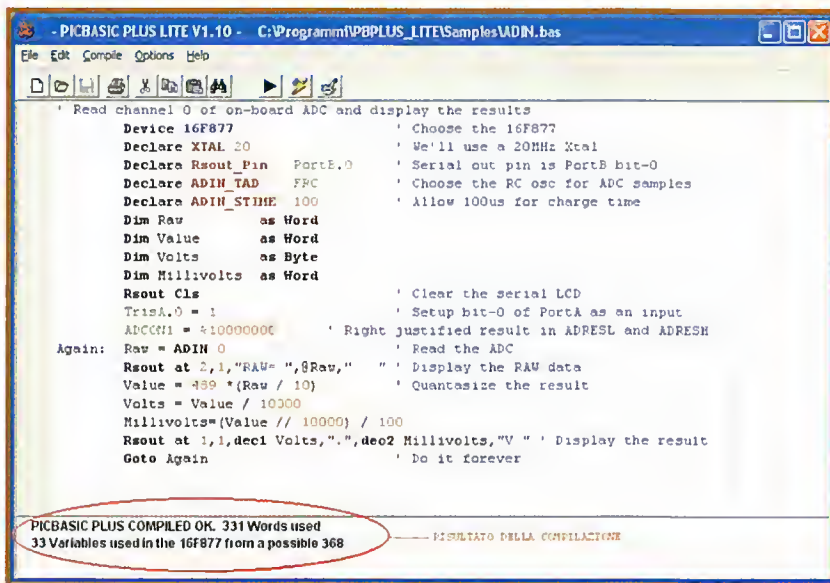
AGAIN: RBB
      Clrw
      F8C rd0ad
      Mov8wz RAW
      Mov8wz PP2H
      Mov8wz RAUH
      B8f1 BPFH,6
      Movlw 192
      F8C curs
      Movlw 'R'
      F8C Rscout
      Movlw 'A'
      F8C Rscout
      Movlw 'V'
      F8C Rscout
      Movlw '='
      F8C Rscout
      Movlw ' '
      F8C Rscout
      Mov8wz RAUH
      Mov8wz PP2H
      Mov8wz RAW
      Mov8wz PP2
      F8C o8dec2
      Movlw ' '
      F8C Rscout
      F8C Rscout
      F8C Rscout
      Clr8f BPFH
      Mov8wz RAUH
      Mov8wz PPOH
      Mov8wz RAW

PICBASIC PLUS COMPILED OK. 331 Words used
33 Variables used in the 16F877 from a possible 368
  
```

Conversione da Basic in Assembler.



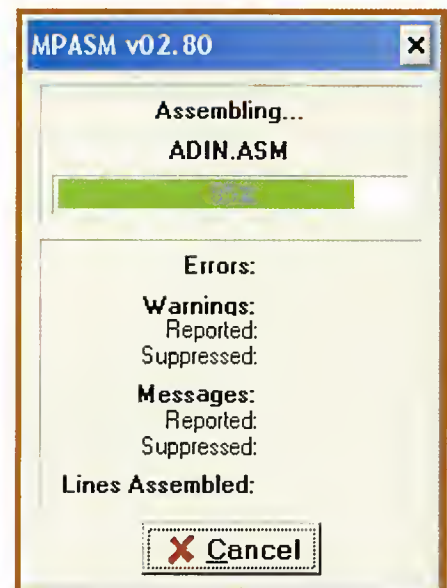
Aiuto del PicBasic Plus Lite.



Il risultato della compilazione è riportato all'interno dell'editor di testo.

in assembler può risultare complicato. Un'istruzione in Basic esegue più operazioni di un'istruzione in assembler.

Osservate ad esempio che in un'istruzione mandiamo il dato sul pin di uscita, inserendo un testo fisso e la variabile da visualizzare, mentre per fare la stessa cosa in assembler avremmo avuto bisogno di molte istruzioni. Nella figura è riportata la conversione eseguita dal programma stesso del file in assembler.



Finestra di stato della compilazione.

Nel manuale di aiuto del programma possiamo vedere come lavorare con le istruzioni di questo ambiente di programmazione, qui viene spiegata l'istruzione e il suo funzionamento e sono riportati esempi di utilizzo. Osservate nella figura il manuale di aiuto nella parte riferita all'istruzione RSOUT.

Compilazione

Per poter vedere la conversione in assembler o in esadecimale (codice macchina) è necessario compilare il programma per fare in modo che generi questi file. Per compilare il programma dobbiamo cliccare sull'apposito pulsante che si trova nella barra degli

strumenti o selezionare l'opzione Compile Basic del menù Compile. In questo modo appare una finestra come quella nella figura in cui è riportato lo stato del processo di compilazione. Quando quest'ultimo si conclude, nella parte inferiore della finestra dell'editor o di lavoro appaiono una serie di informazioni relative ai risultati della compilazione e della dimensione del programma.

Se la compilazione è stata eseguita con successo vengono generati i file da essa risultanti,



```

Code Produced by the
PICBASIC PLUS LITE Compiler, Version 1.10

#include "ADIN.pbp"

RAW      Equ 59
RAWH     Equ 60
VALUE    Equ 61
VALUEH   Equ 62
VOLTS    Equ 63
HILLIVOLTS Equ 64
HILLIVOLTS H Equ 65

B=01 BPFH,6
FBC C1s
Clr0F BPFH
#Define TrisA.0 = 1
Movlw 128
Movwf ADCON1
AGAIN:  RBB
        Clrw
        FBC rdad
        Movwf PAW
        Movwf PP7H
        Movwf RAWH
        B=01 BPFH,6
        Movlw 192
        FBC curs
        Movlw 'R'
        FBC Rscut
        Movlw 'A'
        FBC Rscut
  
```

Hex File Listing

```

0000:006401B9018A28B17BA30FE205B3001
0008:205B307500A930302875178A008230FE
0010:205B0832285B01AD01A715B9300500A6
0018:302700AB30102028300300AB30B82028
0020:01AB3064202801AB300A2028082C282F
0028:00AA082D00A9082C00A82084082800A8
0030:03A6190311B9082719032839022 61803
0038:000808281B0311B919B90008285A00B8
0040:00AA168310061283300900AB1003204F
0048:0CAA0BAB28471403204F083828B61803
0050:285400641406285610062856300000A9
0058:306628753E301B3A283F006400A80DAE
  
```

Finestra del codice
in Assembler e in linguaggio macchina.

quindi potremo selezionare nel menù Options la rappresentazione del codice in assembler (ASM) e/o codice macchina (HEX).

Errori più comuni

Nel programma di esempio che abbiamo presentato ovviamente non esiste alcun errore, ma che cosa accade in presenza di qualche errore all'interno del codice? Come possiamo trovare gli errori?

Inseriamo degli errori all'interno del codice per analizzare ciò che succede. Nella figura abbiamo tolto una lettera all'interno del nome di un registro. Notate che la parola cambia di colore, dato che il compilatore cessa di riconoscerla come "speciale". Compilando il programma appare un messaggio in cui il compilatore ci informa che ci sono diversi errori di compilazione (Too many COMPILER ERRORS), però non dice la posizione dell'errore. Se l'errore lo inseriamo nella definizione di una va-

```

Read channel 0 of on-board ADC and display the results
Device 16F877
Declare XTAL 20
Declare Rscut_Pin PortB.0
Declare ADIN_TAD FPC
Declare ADIN_STIME 100
Dim Raw as Word
Dim Value as Word
Dim Volts as Byte
Dim Hillivolts as Word
TrisA.0 = 1
ADCON1 = %10000000
Again: Raw = ADIN 0
Rscut at 2,1,"RAW= ",8Raw,"
Value = 489 * (Raw / 10)
Volts = Value / 10000
Hillivolts=(Value // 10000) / 100
Rscut at 1,1,dec1 Volts,".",dec2 Hillivolts,"V "
Goto Again
  
```

Too many COMPILER ERRORS.

Inseriamo un errore nel nome del registro.

riabile, così come possiamo vedere nella figura a fianco, al momento della compilazione il messaggio ottenuto è diverso. In questo caso il compilatore ci dice il numero della linea in cui è stato rilevato l'errore e che tipo di errore ha trovato. Nella linea 18 manca l'assegnazione del risultato all'operazione, ovvero non è stata definita correttamente la variabile e quindi il risultato dell'operazione si assegna a "Millivolts" e questo nome non è riconosciuto dal compilatore. Provate altri possibili errori per familiarizzare con il programma e con i messaggi del compilatore. Caricate gli altri programmi di esempio ed esercitatevi con essi prima di iniziare a utilizzare questo software per applicazioni più complesse.

```

Read channel 0 of on-board ADC and display the results
Device 16F877
Declare XTAL 20
Declare Rscut_Pin PortB.0
Declare ADIN_TAD FPC
Declare ADIN_STIME 100
Dim Raw as Word
Dim Value as Word
Dim Volts as Byte
Dim Hillivolts as Word
TrisA.0 = 1
ADCON1 = %10000000
Again: Raw = ADIN 0
Rscut at 2,1,"RAW= ",8Raw,"
Value = 489 * (Raw / 10)
Volts = Value / 10000
Hillivolts=(Value // 10000) / 100
Rscut at 1,1,dec1 Volts,".",dec2 Hillivolts,"V "
Goto Again
  
```

Error at Line (18) in file [ADIN.bas] MILLIVOLTS=(VALUE // 10000) / 100 *** Assignment operator '=' missing! ***
Too many COMPILER ERRORS.

Inseriamo un errore nella definizione di una variabile.



Esercizio: visualizzazione di messaggi, il programma (II)

Abbiamo ancora in sospeso la simulazione dell'esercizio di visualizzazione di diversi messaggi "Addio.asm".

Anche se non abbiamo ancora

il display LCD per provare gli esercizi possiamo simularli e analizzarne la risposta, verificando che il programma risponda all'enunciato iniziale.

Simulazione del programma "Addio.asm"

Abbiamo visto come programmare il PIC per fare in modo che l'LCD visualizzi diversi messaggi sviluppando il codice "Addio.asm" e compilandolo alla ricerca di errori. Ora dobbiamo simularne il funzionamento del programma mediante MPLAB.

Apriamo il progetto "Addio.pjt" e il codice associato per visualizzarlo sul display. Apriremo anche la finestra dei registri speciali e una finestra per vedere i registri più importanti dell'applicazione. Presenteremo la porta B e la variabile Temporale_2 in ASCII, il registro di lavoro, la porta A e la variabile Temporale_1 in binario e le variabili dei temporizzatori in decimale (Delay_Cont, LCD_Temp_1 e LCD_Temp_2). Nella figura potete vedere la finestra che abbiamo descritto.

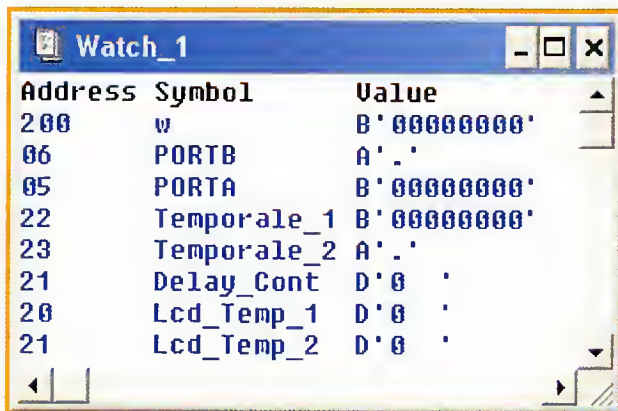
Inizieremo la simulazione passo a passo premendo F7. Come era già successo quando abbiamo simulato il primo esercizio di gestione dell'LCD, quando nel programma principale si esegue la chiamata alla subroutine LCD_INI, si salta alla libreria "lcd_cxx.inc" e si

eseguono le sue istruzioni. Per uscire dalla subroutine LCD_DELAY terremo premuto F7 fino a quando le due variabili temporali saranno 0 o apriremo la finestra Modify e cambieremo il valore delle due variabili, come abbiamo fatto nell'esercizio precedente. Ricordate che dobbiamo ripetere questa operazione tre volte.

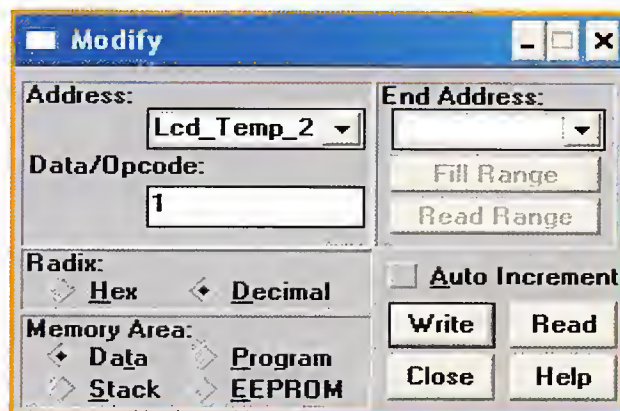
Dopo che il simulatore è ritornato nel programma principale ed è entrato nell'esecuzione del ciclo (Loop) potremo osservare che la porta B sta spedendo correttamente il primo messaggio (ciao) carattere per carattere. Fatto questo, il programma entra nella routine di temporizzazione del TMR0 e rimane in attesa che il flag del temporizzatore si attivi. Forzeremo questo valore a 1 mediante la finestra Modify, come potete vedere dall'immagine in basso e usciremo dalla routine.

Dopo questa attesa simulata il programma continuerà l'esecuzione visualizzando ora il secondo messaggio (Addio). Terminata questa visualizzazione si entrerà nuovamente nella routine di temporizzazione e dopo esserne usciti, il ciclo si ripeterà dall'inizio.

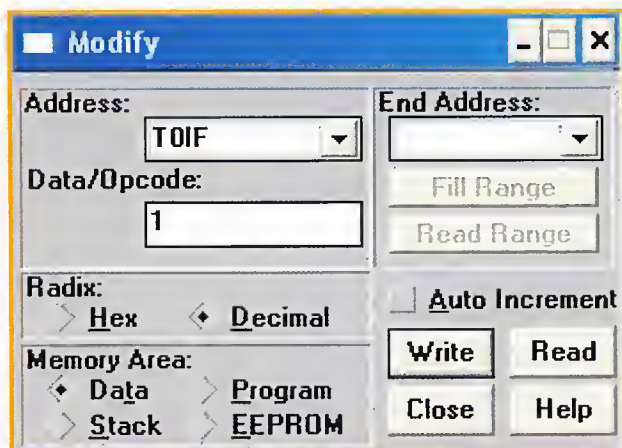
Abbiamo verificato che il programma fun-



Finestra per visualizzare i registri più importanti.



Forziamo il valore delle variabili di temporizzazione.



Forziamo il valore di TOIF per uscire dalla routine.

zione come ci aspettavamo, quindi dobbiamo solamente verificarne il funzionamento sul laboratorio, eseguendo il corrispondente montaggio hardware.

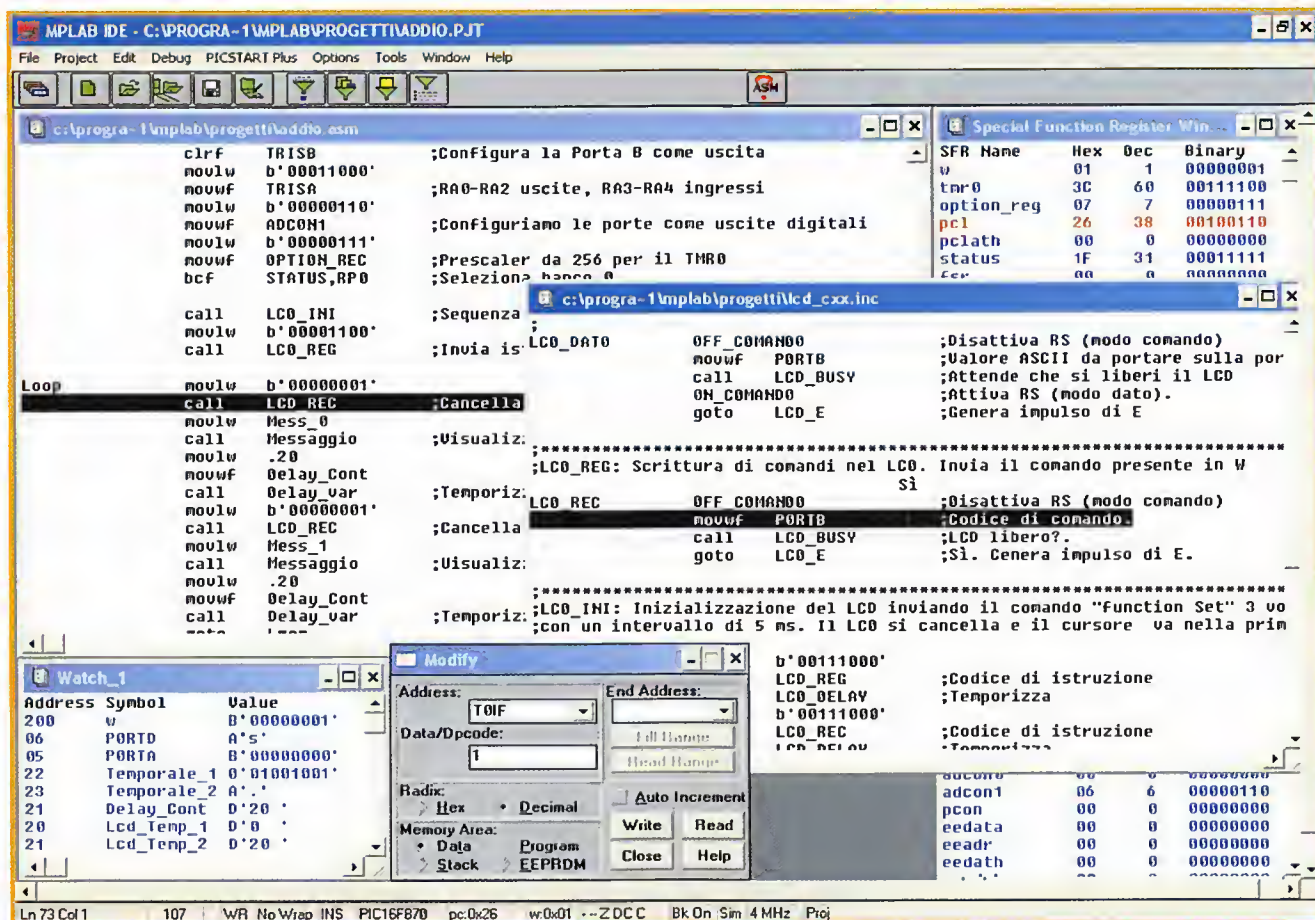
Modifiche al programma

Prima di affrontare nuovi argomenti di programmazione, lavoriamo sul programma che abbiamo realizzato presentando diverse alternative.

Nuovo messaggio

Nel codice "Addio.asm" si contempla solamente la possibilità di visualizzare due messaggi, ma che modifiche dovremmo eseguire per poter visualizzare più messaggi?

Per visualizzare un messaggio nuovo la prima cosa da fare è inserire questo nella tabella dei messaggi. Mantenendo la struttura dei precedenti, inseriamo un nuovo messaggio, ad esempio con il testo "Allarme". L'immagine nella figura della pagina successiva vi permette di verificare come viene inserito il codice corrispondente al nuovo messaggio nella



MPLAB durante la simulazione.



```
addio - Blocco note
File Modifica Formato Visualizza ?
retlw 0x00 ;Ultimo carattere del messaggio 0
Mess_1
equ $ ;Mess_1 punta al primo carattere del messaggio 1
retlw 'A'
retlw 'd'
retlw 'i'
retlw 'o'
retlw 0x00 ;Ultimo carattere del messaggio 1
Mess_2
equ $ ;Mess_2 punta al primo carattere del messaggio 2
retlw 'L'
retlw 'i'
retlw 'a'
retlw 'r'
retlw 'm'
retlw 'e'
retlw 0x00 ;Ultimo carattere del messaggio 2
*****
;Delay_var: Questa routine di carattere generale realizza una temporizzazione variabile
;fra 50 ms e 12.8". Si utilizza un prescaler da 256 e si carica il TMRO con 195.
```

Modifichiamo la tabella inserendo un nuovo messaggio.

```
addio - Blocco note
File Modifica Formato Visualizza ?
movlw .20
movwf Delay_Count
call Delay_var ;Temporizza 2 secondi
movlw b'00000001'
call LCD_REG ;Cancella LCD y Home (colloca il cursore nella 1ª posizione)
call Mess_1 ;Visualizza il Messaggio 1
movlw .20
movwf Delay_Count
call Delay_var ;Temporizza 2 secondi
movlw b'00000001'
call LCD_REG ;Cancella LCD y Home (colloca il cursore nella 1ª posizione)
movlw Mess_2 ;Visualizza il Messaggio 2
movlw .20
movwf Delay_Count
call Delay_var ;Temporizza 2 secondi
goto Loop
end ;Fine del programma sorgente
```

Modifichiamo il programma per far visualizzare il nuovo messaggio.

tabella dei messaggi. Possiamo vedere il nuovo codice evidenziato in azzurro.

Il nuovo messaggio è ora all'interno della tabella ma non abbiamo ancora modificato il programma che visualizza i messaggi. Se andiamo al ciclo di visualizzazione, a partire dal-

l'etichetta Loop possiamo verificare che ogni messaggio che viene visualizzato ha delle istruzioni associate. Queste iniziano con l'istruzione tramite la quale carichiamo sul registro di lavoro il valore '00000001' per chiamare la routine "LCD_REG" in modo da cancellare il display e collocare il cursore nella prima posizione. Il blocco di istruzioni associate a ogni messaggio termina quando chiamiamo la routine di temporizzazione "call Delay_var" per temporizzare i 2 secondi.

Se copiamo questo blocco di istruzioni e lo adattiamo per il nuovo messaggio che vogliamo visualizzare, otterremo la visualizzazione sul display LCD dei nostri tre messaggi.

Notate dove viene inserito il blocco di istruzioni (dopo quelle che corrispondono al secondo messaggio) e come è stato adattato per il nostro messaggio (movlw Mess_2).

Fatto questo, dobbiamo salvare le modifiche che abbiamo realizzato nel programma memorizzandolo con un nome diverso, ad esempio "Addio1.asm". Il passo successivo è compilare il codice per verificare che le modifiche non diano errori.

```
Build Results
Building ADDIO1.HEX...

Compiling ADDIO1.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /p16F870 /q C:\PROGRA~1\MPLAB\PROGETTI\ADDIO1.ASM"
Message[302] C:\PROGRA~1\MPLAB\PROGETTI\ADDIO1.ASM 90 : Register in operand not in bank 0. Ensure
Message[302] C:\PROGRA~1\MPLAB\PROGETTI\ADDIO1.ASM 92 : Register in operand not in bank 0. Ensure
Message[302] C:\PROGRA~1\MPLAB\PROGETTI\ADDIO1.ASM 94 : Register in operand not in bank 0. Ensure
Message[302] C:\PROGRA~1\MPLAB\PROGETTI\ADDIO1.ASM 96 : Register in operand not in bank 0. Ensure

Build completed successfully.
```

Risultato della compilazione.



Cambi nella temporizzazione

Immaginate ora di volere che la temporizzazione dei tre messaggi non sia uguale e che il nuovo messaggio si veda la metà del tempo degli altri. Utilizzando la stessa routine potremo temporizzare tra 50 millisecondi e 12,8 secondi. Dovremo semplicemente cambiare il valore della variabile Delay_cont e inserire in essa il fattore da far moltiplicare con i 50 ms di ritardo minimo che genera la subroutine. Se inseriamo 10 invece di 20, che è il valore degli altri due messaggi, otterremo che il nuovo messaggio si visualizzi la metà del tempo degli altri due. Nell'immagine della figura possiamo vedere la modifica sull'istruzione che carica il registro di lavoro con il valore che successivamente verrà dato alla variabile.

```

addio1 - Blocco note
File Modifica Formato Visualizza 2
call    Messaggio1      ;visualizza il Messaggio 1
movlw   .20              ;Temporizza 2 secondi
movwf   Delay_cont
call    Delay_var

movlw   b'00000001'     ;Cancella LCD e Home (colloca il cursore nella 1ª posizione)
call    LCD_REG
movlw   Mess_2
call    Messaggio       ;visualizza il Messaggio 2
movlw   .10              ;Temporizza 1 secondi
movwf   Delay_cont
call    Delay_var

goto    Loop
end                      ;Fine del programma sorgente

```

Cambiamo la temporizzazione della visualizzazione.

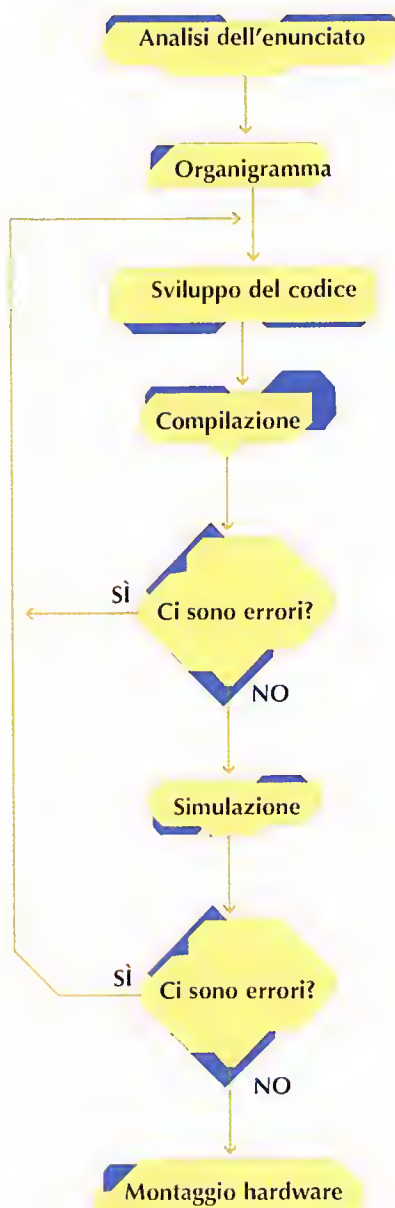
Conclusioni

Qualsiasi variazione nel programma richiede una nuova compilazione e simulazione. Dobbiamo prendere come regola quella di compilare e simulare il programma modificato prima di procedere alla scrittura sul PIC.

Avete potuto verificare con quale semplicità si possono fare variazioni in un programma per adattarlo alle nuove necessità. Questo è un grande vantaggio al momento di sviluppare nuove applicazioni, dato che normalmente potremo utilizzare programmi precedentemente risolti in modo che servano come base per lo sviluppo di quelli nuovi.

Nel programma di visualizzazione con display LCD "Addio.asm" abbiamo predisposto due semplici modifiche tra tutte quelle che si sarebbero potute fare, ma le possibilità dei programmi dipendono dall'immaginazione che ha il programmatore. Ad esempio, sarà possibile predisporre un programma che visualizzi un messaggio in funzione dell'ingresso selezionato, inoltre è possibile incatenare messaggi in funzione degli ingressi, ecc.

Nei programmi successivi complicheremo un po' di più le applicazioni di gestione dell'LCD in modo da affinare le conoscenze e il metodo di lavoro per utilizzare questo dispositivo esterno.



Passaggi raccomandati per sviluppare un'applicazione.